设计文档

1.数据通路

本流水线CPU为五级流水线结构。功能部件规划如下：F(IFU), D(NPC, RF, EXT, CMP), E(ALU), M(DM), W(RF)。

1.1.分布式译码

由F_cu，D_cu，E_cu，M_cu，W_cu对当前流水段进行译码分析。以下是CU的说明：

CU

端口说明

信号名称	方向	描述
opcode	I	Instr[31:26]
func	I	Instr[5:0]
NPCOp	O	NPC的控制信号
CMPOp	O	CMP的控制信号
RFWrEn	O	RF的写使能
RFWRSel	O	RF写寄存器的选择信号
RFWDSel	O	RF写数据的控制信号
EXTOp	O	EXT的控制信号
ALUOp	O	ALU的控制信号
ALUASel	O	ALUA的选择信号
ALUBSel	O	ALUB的选择信号
DMWrEn	O	DM的写使能
DMOp	O	DM的控制信号
DMEXTOp	O	DM的符号扩展控制信号
J_l	O	指令为“跳转-链接”型
Calc_r	O	指令为“ALU计算&R”型
Calc_i	O	指令为“ALU计算&I”型
Lui	O	指令为“Lui”型
Load	O	指令为“Load”型
Store	O	指令为“Store”型
Branch	O	指令为“Branch”型
Jr	O	指令为“跳转至寄存器型”

真值表

lui的结果产生移至D级的EXT，E_E32即结果。利用ALU的ADD和GRF[0]相加，在M级和M_AO合流。

	NPCOp[3:0]	RFWrEn	RFWRSel[2:0]	RFWDSel[2:0]	EXTOp	ALUOp[3:0]	CMPOp[3:0]	ALUASel[1:0]	ALUBSel[1:0]	DMWrEn	DMOp[1:0]	DMEXTOp
add	NPCOp_PC4	1	RFWRSel_rd	RFWDSel_ALU		ADD	CMPOp_non	ALUASel_rs	ALUBSel_rt			
sub	NPCOp_PC4	1	RFWRSel_rd	RFWDSel_ALU		SUB	CMPOp_non	ALUASel_rs	ALUBSel_rt			
ori	NPCOp_PC4	1	RFWRSel_rt	RFWDSel_ALU	EXTOp_zero	OR	CMPOp_non	ALUASel_rs	ALUBSel_imm			
lui	NPCOp_PC4	1	RFWRSel_rt	RFWDSel_ALU	EXTOp_lui	ADD	CMPOp_non	ALUASel_rs	ALUBSel_imm			
lw	NPCOp_PC4	1	RFWRSel_rt	RFWDSel_DMRD	EXTOp_sign	ADD	CMPOp_non	ALUASel_rs	ALUBSel_imm		DMOp_w	
sw	NPCOp_PC4	0			EXTOp_sign	ADD	CMPOp_non	ALUASel_rs	ALUBSel_imm	1	DMOp_w	
jal	NPCOp_JL	1	RFWRSel_31	RFWDSel_PC8			CMPOp_non	ALUASel_rs				
jr	NPCOp_JR	0					CMPOp_non	ALUASel_rs				
beq	NPCOp_Br	0					CMPOp_beq	ALUASel_rs	ALUBSel_rt			

1.2.功能部件

1.2.1.Stall

Stall内部三次实例化CU，对三级指令译码，解出T_{use}和T_{new}。

1.2.1.1.阻塞条件

- $T_{new}>T_{use}$
- 用的寄存器和写的寄存器是同一个，即D_rs_addr==A3
- 用的寄存器不为$0

端口说明

信号名称	方向	描述
D_Instr	I	D级指令
E_Instr	I	E级指令
M_Instr	I	M级指令
E_A3	I	E级指令写的寄存器
M_A3	I	M级指令写的寄存器
Stall	O	高电平暂停

1.2.1.2.暂停实现

在D级暂停，加入气泡。因此要给ifu和FD_REG, DE_REG加上使能信号。DE_REG中：当使能信号为低电平时，将E_Instr赋值为nop。

需求者的最晚时间模型·T_{use}

T_{use}：指令进入D级后，其后的某个功能部件再经过多少cycle就**必须**要使用寄存器值

供给者的最早时间模型·T_{new}

T_{new}：位于E级及其后各级的指令，再经过多少cycle能够产生要写入寄存器的结果。

暂停转发的基本方法

A指令位于D级，将A的T_{use}与位于E/M/W各级指令的T_{new}比较。若T_{new}大，则暂停，否则转发。

暂停机制构造方法

注意事项

- 只关注每条指令的操作语义
- 指令可能有2个不同的T_{use}

真值表

- T_{use}：无定义处应取极限大值。避免stall误触发。

指令	T_{use}^{rs}	T_{use}^{rt}
Calc_r	1	1
Calc_i	1	3
Lui	0	0
Load	1	3
Store	1	2
Branch	0	0
Jr	0	3

- T_{new}：无定义处应取0。避免stall误触发。

指令类型	T_{new}@E	T_{new}@M	T_{new}@W
Calc_r	1	0	0
Calc_i	1	0	0
Lui	0	0	0
Load	2	1	0
Store	0	0	0
Branch	0	0	0
J_l	0	0	0

1.2.2.F_IFU

端口说明

信号名称	方向	描述
WE	I	写使能(!Stall)
clk	I	时钟信号
reset	I	同步复位
NPC	I	来自NPC的次地址
Instr	O	指令
PC	I	更新后的地址

1.2.3.FD_REG

端口说明

信号名称	方向	描述
F_Instr	I	F级指令
F_PC	I	F级地址
D_Instr	O	D级指令
D_PC	O	D级地址
clk	I	时钟信号
reset	I	同步复位
WE	I	写使能(!Stall)
flush	I	高电平时清空延迟槽。flush=CMPOp==`CMPOp_Bxxzall && !D_b_jump

1.2.4.D_CU

控制NPC, RF, EXT, CMP。

需要译码得到RFWRSel，以此确定A3。当得到了写寄存器的编号，就可以判断转发。所以A3参与流水。

WD不参与流水，因为它具有不确定性，故让其MUX数据集进入流水。

端口说明

信号名称	方向	描述
opcode	I	D_Instr[31:26]
func	I	D_Instr[5:0]
NPCOp	O	NPC的控制信号
RFWRSel	O	RF-A3的选择信号
EXTOp	O	EXT的控制信号
CMPOp	O	CMP的控制信号

1.2.5.D_NPC

B指令和**J指令**支持延迟槽，所以使用PC@F+4。此次设计不采用直接输出PC8的格式，而是在PC的流水中，随用随加8。

同时接受F_PC和D_PC的输入。当D指令的NPCOp为顺序default值时，NPC=**F**_PC+4；当D指令的NPCOp为B类时，NPC采用PC@**F**+4；当D指令为JL类，NPC只与**D**相关；最后当D指令为JR类时，NPC=RS。

输入RS被转发处理过。

NPC输出**直接回写F级。**

端口说明

信号名称	方向	描述
F_PC	I	F级指令
NPCOp	I	D级指令对应的NPC控制信号
IMM26	I	26位立即数
IMM16	I	16位立即数
RS	I	jr需要的寄存器内容
Branch	I	B指令跳转选择
NPC	O	回写F级IFU

控制信号

NPCOp	描述
NPC_PC4	顺序
NPC_Br	分支
NPC_JL	跳转并链接
NPC_JR	跳转寄存器内容

1.2.6.D_RF

只实例化一次，D级只使用写功能。读功能的部署在W级。

- **内部转发**：当读和写同一个寄存器时，**读出的数据应该为要写入的数据。**
- 输出D_V1和D_V2**经过转发处理，或许进入流水。**

端口说明

信号名称	方向	描述
clk	I	时钟信号
reset	I	同步复位
A1	I	rs寄存器编号，D_Instr[25:21]
A2	I	rt寄存器编号，D_Instr[20:16]@D
RD1	O	D_V1
RD2	O	D_V2
W_Instr	I	W级指令
RFWrEn	I	W级指令控制
A3	I	W_A3
WD	I	W_RW

1.2.7.D_EXT

输出D_E32进入流水。

端口说明

信号名称	方向	描述
IMM16	I	16位立即数
EXTOp	I	EXT的控制信号
IMMEXT	O	32位扩展立即数D_E32 / lui结果

控制信号

EXTOp	描述
EXTOp_zero	0扩展
EXTOp_sign	符号扩展
EXTOp_lui	lui计算

1.2.8.D_CMP

端口说明

信号名称	方向	描述
D1	I	经过转发处理的GRF[rs]，即FD_rs
D2	I	经过转发处理的GRF[rt]，即FD_rt
CMPOp	I	CMP的控制信号
Branch	O	NPC的跳转信号

控制信号

CMPOp	描述
CMPOp_bep	beq比较

1.2.9.DE_REG

输出E_A3进入流水，E_V2经过转发处理，或许进入流水。

控制信号

信号名称	方向	描述
WE	I	写使能，默认为1
flush		阻塞时插入nop->flush=Stall
D_b_jump	I	D级B指令是否跳转
E_b_jump	O	
D_V1	I	经过转发处理的GRF[rs]，即FD_rs
D_V2	I	经过转发处理的GRF[rt]，即FD_rt
D_E32	I	32位扩展立即数 / **lui的计算结果**
D_Instr	I	指令
D_PC	I	地址
D_A3		写寄存器编号
E_Instr	O	
E_PC	O	
E_V1	O	GRF[rs]
E_V2	O	GRF[rt]
E_E32		32位扩展立即数 / **lui的计算结果**
E_A3		
clk	I	时钟信号
reset	I	同步复位

1.2.10.E_ALU

输出AO进入流水。

端口说明

信号名称	方向	描述
A	I	经过转发处理和MUX选择的操作数A
B	I	经过转发处理和MUX选择的操作数B
ALUOp	I	ALU的控制信号
AO	O	ALU的结果

控制信号

ALUOp	描述
ADD	加法
SUB	减法
OR	或运算
LUI	加载立即数至高位

1.2.11.EM_REG

端口说明

信号名称	方向	描述
E_b_jump	I	E级B指令是否跳转
M_b_jump	O	
E_V2	I	经过转发处理的GRF[rt]，即FE_rt
E_AO	I	ALU的结果
E_Instr	I	指令
E_PC	I	地址
E_A3	I	写寄存器编号
M_V2	O	
M_AO	O	
M_Instr	O	
M_PC	O	
M_A3	O	
clk	I	时钟信号
reset	I	同步复位

1.2.12.M_DM

端口说明

信号名称	方向	描述
M_PC	I	地址
WE	I	写使能信号
A	I	ALU计算出的地址（M_AO）
WD	I	经过转发处理的WD，即FM_DW
DMEXTOp	I	扩展控制信号
DMOp	I	DM的控制信号
DMRD	O	DM读出的数据
clk	I	时钟信号
reset	I	同步复位

控制信号

DMOp[1:0]	描述
`DMOp_w`	以字为单位读/写
`DMOp_h`	以半字为单位读/写
`DMOp_b`	以字节为单位读/写

DMEXTOp	描述
`DMEXTOp_zero`	零扩展
`DMEXTOp_sign`	符号扩展

1.2.13.MW_REG

端口说明

信号名称	方向	描述
M_b_jump	I	M级B指令是否跳转
W_b_jump	O	
M_AO	I	ALU计算出的结果
M_DR	I	DM读出的结果
M_Instr	I	指令
M_PC	I	地址
M_A3	I	写寄存器编号
W_AO	O	
W_DR	O	
W_Instr	O	
W_PC	O	
W_A3		
clk	I	时钟信号
reset	I	同步复位

1.2.14.W_CU

端口说明

信号名称	方向	描述
opcode	I	W_Instr[31:26]
func	I	W_Instr[5:0]
RFWrEn	O	RF的写使能
RFWDSel	O	RF的写数据选择

1.2.15.Forward

外部转发：mips转发区（FD_rs，FD_rt，FE_rs，FE_rt，FM_DW）

内部转发：RF寄存器内部。当读写寄存器相同且编号不为0时进行内部转发。

需求者

- RF内部读写编号相同时的输出端口（内部转发：RFWD->RFRD）
- CMP的两个输入端口
- ALU的两个输入端口
- NPC的RA输入端
- DM的写数据输入端
- DE_REG, EM_REG传递的寄存器值

选择数据

借用mips.v中的cu进行译码，确定供给者是否转发（信号：Forward）。

采用**就近原则**选择级次转发的数据。

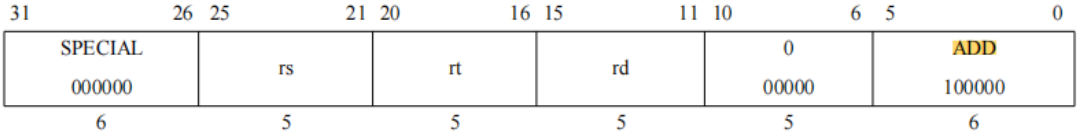
转发输出(优先级：高->低)	恰在本级产生寄存器结果的指令	供给者
DE_REG	J_L / LUI	E_PC+32'd8 / E_E32
EM_REG	CALC_R / CALC_I	M_PC+32'd8 / M_AO
MW_REG	LOAD	W_RW
寄存器内部转发		

接受条件

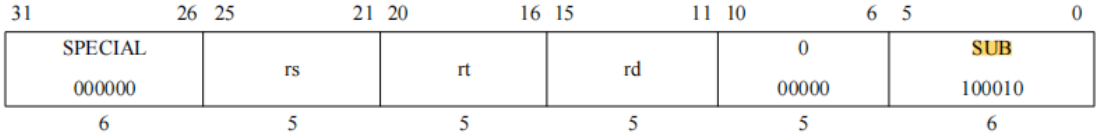
- 供给端的寄存器地址与当前的相同
- 当前需要的地址不为0
- 供给端可以转发 Forward

2.指令

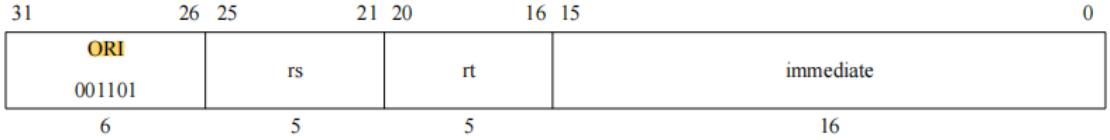
- add



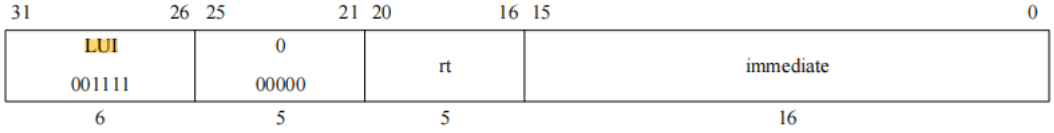
- sub



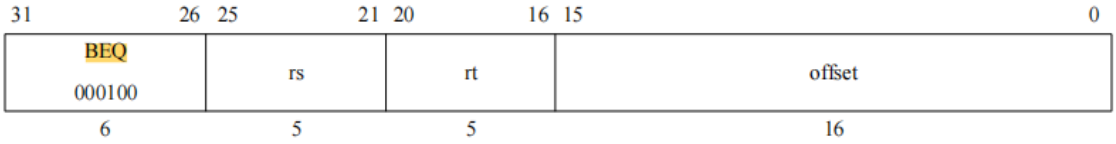
- ori



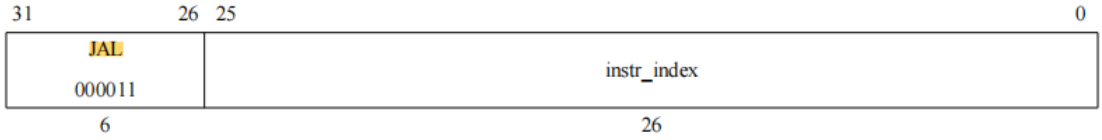
- lui



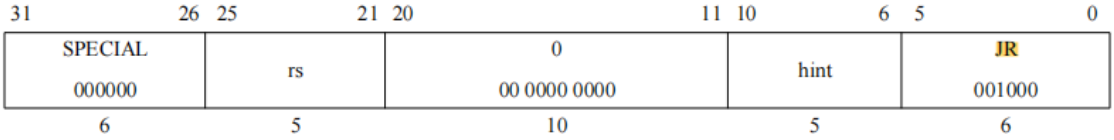
- beq



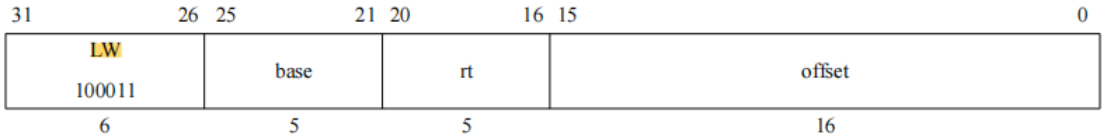
- jal



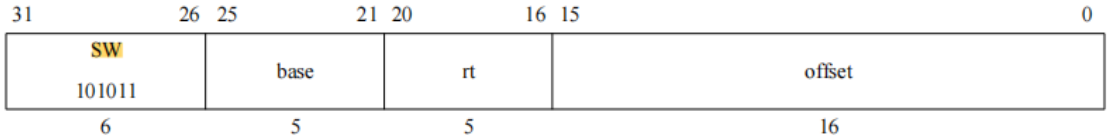
- jr



- lw



- sw



3.测试方案

自动化工具与P4保持一致。

数据生成器方面，我使用了上次P4的数据生成器做基本测试。同时手动模拟了一些冒险阵列。

此外，因时间不够所以使用了他人的强度较高的数据生成器进行测试。

4.思考题

1.我们使用提前分支判断的方法尽早产生结果来减少因不确定而带来的开销，但实际上这种方法并非总能提高效率，请从流水线冒险的角度思考其原因并给出一个指令序列的例子。

提前判断使得b指令的T_{use}为0，增大出现$T_{new}>T_{use}$的概率。从而阻塞概率升高。

```
ori $a0, $0, 1
ori $a1, $0, 2
beq $a0, $a1, target
```

2.因为延迟槽的存在，对于 jal 等需要将指令地址写入寄存器的指令，要写回 PC + 8，请思考为什么这样设计？

因为当jal在D级时，F级已经完成取指，延迟槽必定进入流水。写回PC+8才可以使指令没有重复地被高效运行。

3.我们要求所有转发数据都来源于流水寄存器而不能是功能部件（如 DM、ALU），请思考为什么？

功能部件的输出有延迟，故产生的数据并不稳定，而流水寄存器中的数据来自前一级已经计算出来的数据，可以在当前数据稳定输出。

4.我们为什么要使用 GPR 内部转发？该如何实现？

当读寄存器和写寄存器相同时，为了等待时钟上升沿的到来，写操作延迟进行，故当前周期指令并不能取到最新的数据。

实现：当读寄存器和写寄存器相同，写使能为高电平，寄存器编号不为0时，将写数据转发给读数据。

5.我们转发时数据的需求者和供给者可能来源于哪些位置？共有哪些转发数据通路？

- 需求者
  - 内部转发：grf内部RD1/RD2
  - cmp的两个输入端口
  - npc的rs端口
  - alu的两个输入mux的寄存器端
  - dm的写数据端口
  - DE_REG, EM_REG传递的寄存器值
- 供给者
  - 内部转发：grf的WD
  - DE_REG的E_E32和E_PC+32'd8
  - EM_REG的M_AO和M_PC+32'd8
  - MW_REG的W_RW
- 转发数据通路
  - 内部转发
    - grf：WD~~>RD1 / RD2
  - lui
    - DE_REG的E_E32~~>FD_rs / FD_rt
    - EM_REG的M_AO~~>FD_rs / FD_rt / FE_rs / FE_rt
    - MW_REG的W_RW~~>FD_rs / FD_rt / FE_rs / FE_rt / FM_DW
  - jal
    - DE_REG的E_PC+32'd8~~>FD_rs / FD_rt
    - EM_REG的E_PC+32'd8~~>FD_rs / FD_rt / FE_rs / FE_rt
    - MW_REG的W_RW~~>FD_rs / FD_rt / FE_rs / FE_rt / FM_DW
  - cal_r / cal_i
    - EM_REG的M_AO~~>FD_rs / FD_rt / FE_rs / FE_rt

        - MW_REG的W_RW~~>FD_rs / FD_rt / FE_rs / FE_rt / FM_DW
    - load
        - MW_REG的W_RW~~>FD_rs / FD_rt / FE_rs / FE_rt / FM_DW

6.在课上测试时，我们需要你现场实现新的指令，对于这些新的指令，你可能需要在原有的数据通路上做哪些扩展或修改？提示：你可以对指令进行分类，思考每一类指令可能修改或扩展哪些位置。

- 数据通路：根据CU的信号，写出真值表。以真值表为索引修改数据通路
- 暂停和转发：写出AT矩阵，**将新指令分类写入CU的指令类型模块**。根据矩阵和分类修改stall.v和forward的内容。

7.简要描述你的译码器架构，并思考该架构的优势以及不足。

分布式译码。由F_cu，D_cu，E_cu，M_cu，W_cu对当前流水段进行译码分析。

优点：各司其职，分工明确。

缺点：较为分散，管理相对费力。